# 11. Creating a Basic CLI Application

Write a Ruby script to create a basic command-line interface (CLI) application using `optparse`.

```
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ./11-cli.rb -h
Usage: cli.rb [options]
    -a, --add TASK                   Add a new task
    -l, --list                       List all tasks
    -r, --remove INDEX               Remove a task by index
    -h, --help                       Show help
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ./11-cli.rb -a Task1
Task 'Task1' added.
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ cat tasks.txt
Task1
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ./11-cli.rb -a Task2
Task 'Task2' added.
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ./11-cli.rb -l
1. Task1
2. Task2
 ┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ./11-cli.rb -r 2
Task 'Task2' removed.
```

### Script: `11-cli.rb`

```ruby
require 'optparse'

# File to store tasks
TASKS_FILE = 'tasks.txt'

# Load tasks from the file or initialize an empty array
def load_tasks
  if File.exist?(TASKS_FILE)
    File.readlines(TASKS_FILE).map(&:chomp)
  else
```

```ruby
    []
  end
end

# Save tasks to the file
def save_tasks(tasks)
  File.open(TASKS_FILE, 'w') do |file|
    tasks.each { |task| file.puts(task) }
  end
end

# Add a new task
def add_task(task)
  tasks = load_tasks
  tasks << task
  save_tasks(tasks)
  puts "Task '#{task}' added."
end

# List all tasks
def list_tasks
  tasks = load_tasks
  if tasks.empty?
    puts "No tasks found."
  else
    tasks.each_with_index do |task, index|
      puts "#{index + 1}. #{task}"
    end
  end
end

# Remove a task by index
def remove_task(index)
  tasks = load_tasks
  if index < 1 || index > tasks.size
    puts "Error: Invalid task index."
  else
    removed_task = tasks.delete_at(index - 1)
    save_tasks(tasks)
    puts "Task '#{removed_task}' removed."
  end
end
```

```
# CLI options parsing
options = {}
OptionParser.new do |opts|
  opts.banner = "Usage: cli.rb [options]"

  opts.on('-a', '--add TASK', 'Add a new task') do |task|
    options[:add] = task
  end

  opts.on('-l', '--list', 'List all tasks') do
    options[:list] = true
  end

  opts.on('-r', '--remove INDEX', Integer, 'Remove a task by index') do
|index|
    options[:remove] = index
  end

  opts.on('-h', '--help', 'Show help') do
    puts opts
    exit
  end
end.parse!

# Handle options
if options[:add]
  add_task(options[:add])
elsif options[:list]
  list_tasks
elsif options[:remove]
  remove_task(options[:remove])
else
  puts "Usage: cli.rb [options]"
  puts "Run with -h for help."
end
```

---

### Explanation of the Script:

1. **Task Management**:
   - **File-based Storage**: Tasks are stored in a `tasks.txt` file for persistence.
   - **Load and Save**: Functions `load_tasks` and `save_tasks` handle reading and writing tasks to the file.
2. **Options**:

    - `-a` or `--add TASK`: Adds a new task.
    - `-l` or `--list`: Lists all tasks with their indices.
    - `-r` or `--remove INDEX`: Removes a task by its index.
    - `-h` or `--help`: Displays help and usage instructions.
3. **Error Handling**:
    - Ensures a valid index is provided for removing tasks.
    - Displays a helpful message if no options are passed.
4. **CLI Parsing**:
    - The `OptionParser` library is used to parse command-line arguments and execute corresponding actions.

---

### Example Usage:

1. **Help**:

```
$ ruby 11-cli.rb -h
Usage: cli.rb [options]
    -a, --add TASK                   Add a new task
    -l, --list                       List all tasks
    -r, --remove INDEX               Remove a task by index
    -h, --help                       Show help
```

2. **Add Tasks**:

```
$ ruby 11-cli.rb -a Task1
Task 'Task1' added.

$ ruby 11-cli.rb -a Task2
Task 'Task2' added.
```

3. **List Tasks**:

```
$ ruby 11-cli.rb -l
1. Task1
2. Task2
```

4. **Remove a Task**:

```
$ ruby 11-cli.rb -r 2
Task 'Task2' removed.

$ ruby 11-cli.rb -l
1. Task1
```

---

This script creates a robust CLI application for managing tasks and supports adding, listing, and removing tasks with ease. It also ensures tasks are stored persistently across script runs.